Distributed Graph-Based Entity Resolution Using Spark

Mahdi Namazifar, PhD
Spark Summit East, March 2015
About Me

• Senior Data Scientist with Talos team of Cisco Security Business Group

• Cisco’s San Francisco Innovation Center

• We are building the Big-Data architecture, as well as Analytics and machine learning foundations for network security

• My homepage:  www.namazifar.com
Entity Resolution (ER)
Entity Resolution

• a.k.a. “Record Linkage”
Entity Resolution

• a.k.a. “Record Linkage”

• Same entities could be referred to in different ways
Entity Resolution

• a.k.a. “Record Linkage”

• Same entities could be referred to in different ways
Entity Resolution

- a.k.a. “Record Linkage”
- Same entities could be referred to in different ways

Person:
- Michael Peterson
- Mike Peterson
- Mike H. Peterson
- Michael Henry Peterson

Country:
- USA
- US
- U.S.A.
- United States
- United States of America
- Untd. St. of America
Entity Resolution

• a.k.a. “Record Linkage”
• Same entities could be referred to in different ways
Entity Resolution

“Identifying descriptions of same entity in one or multiple data sources”
Ideas from Entity Resolution Literature
Ideas from Entity Resolution Literature
Ideas from Entity Resolution Literature

String Similarity Measures
Ideas from Entity Resolution Literature

String Similarity Measures

Blocking Algorithms
Ideas from Entity Resolution Literature

String Similarity Measures

Blocking Algorithms

Graphs

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
Standard Representation of Entities
Two Major Classes of ER problems

Class 1: There exists standard representation(s) for entities
Two Major Classes of ER problems

Class 1: There exists standard representation(s) for entities

Non-Standard Representations for a Country

U.S.A.
United States of America
Untd. St. of America
US of A
Two Major Classes of ER problems

Class 1: There exists standard representation(s) for entities

Non-Standard Representations for a Country

- U.S.A.
- United States of America
- Untd. St. of America
- US of A

Standard Representations Countries

- ISO3166-1 alpha-2: US, BE, AT, ...
- ISO3166-1 alpha-3: USA, BEL, AUT, ...
- Standard Names: United States, Belgium, Austria, ...
- Standard Local Names: United States, Belgique, Österreich, ...
Two Major Classes of ER problems

Class 2: There exists **NO** standard representation for entities*

* At least none that could be used for the specific data set
Two Major Classes of ER problems

Class 2: There exists *NO* standard representation for entities*

* At least none that could be used for our specific data set
Two Major Classes of ER problems

Class 2: There exists **NO** standard representation for entities*

* At least none that could be used for our specific data set
Two Major Classes of ER problems

Class 2: There exists **NO** standard representation for entities*

* At least none that could be used for our specific data set
How Does our ER Algorithm Work
Real Example

• Whois data set for .com
Real Example

• Whois data set for .com – Interested in “Registrant Country”
Real Example

- Whois data set for .com – Interested in “Registrant Country”
- 115M records
Real Example

• Whois data set for .com – Interested in “Registrant Country”
• 115M records; 80,000 unique values!
Real Example

• Whois data set for .com – Interested in “Registrant Country”
• 115M records; 80,000 unique values!

Fields indicating registrant country (may or may not be empty)

• Registrant Country
• Registrant Country Code
• Registrant Phone Number
Real Example

- Whois data set for .com – Interested in “Registrant Country”
- 115M records; 80,000 unique values!

Fields indicating registrant country (may or may not be empty)

- Registrant Country
- Registrant Country Code
- Registrant Phone Number

Example

- Czech
- +420-2-2143-5111
Build the Graph

Standard Representation

Non-Standard Representation
Build the Graph

Standard Representation

Non-Standard Representation
Build the Graph

Standard Representation

Non-Standard Representation

<table>
<thead>
<tr>
<th>ID</th>
<th>Country Name</th>
<th>Country Code</th>
<th>Calling Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Build the Graph – Schema
Build the Graph – Example

[Diagram showing connections between United States and Czech Republic with country codes and ID numbers.]

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420
Do We Need Graph for This Example?
Build the Graph – Example

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420
Standard Representation
Standard Representation
Semi-bipartite graph
Standard Representation
Semi-bipartite graph

**BUT** that’s not always the case
Graph Traversal – Paths of length <= 3

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420
Graph Traversal – Paths of length <= 3

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420

United States
US
+1

Czech Republic
CZ
+420

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420
Graph Traversal – Paths of length <= 3
Graph Traversal – Paths of length <= 3

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420
Graph Traversal – Paths of length <= 3

United States

ID: 1002
Country Name: Untd. St. of America
Country Code: US
Calling Code: +1

ID: 1003
Country Name: -
Country Code: USA
Calling Code: +1

ID: 1004
Country Name: -
Country Code: -
Calling Code: +420

Czech Republic

CZ

+420

CZE
Graph Traversal – Algorithm

- **Problem**: find all paths of length $k$ in a graph
Graph Traversal – Algorithm

• **Problem:** find all paths of length $k$ in a graph

• **One Solution** (using message passing):
Graph Traversal – Algorithm

- **Problem:** find all paths of length $k$ in a graph
- **One Solution** (using message passing):  
  - $k$ iterations
Graph Traversal – Algorithm

• **Problem:** find all paths of length $k$ in a graph

• **One Solution** (using message passing):
  
  • $k$ iterations
  
  • **Iteration 1:**
    
    • Each node broadcasts its ID as a message
    
    • Each node receives messages and append its own ID to the end of each message ($\text{msg}_{\text{mod}}$)
Graph Traversal – Algorithm

• **Problem:** find all paths of length $k$ in a graph

• **One Solution** (using message passing):
  
  • $k$ iterations
  
  • Iteration 1:
    
    • Each node broadcasts its ID as a message
    
    • Each node receives messages and append its own ID to the end of each message ($\text{msg}_{\text{mod}}$)
  
  • Iteration $i$ ($1 < i \leq k$):
    
    • Each node broadcasts its $\text{msg}_{\text{mod}}$
    
    • Each node receives messages and append its ID to the end of each message ($\text{msg}_{\text{mod}}$)
Graph Traversal – GraphX

Function for calculating the messages

```scala
def calcMsg(triplet: EdgeTriplet[List[(List[Long], Double)], Double]) : List[(List[Long], Double)] = {
  return (triplet.srcAttr).map(l => (l._1++List(triplet.srcId), math.min(triplet.attr, l._2)))
}
```

Message passing

```scala
val newGraph = graph.pregel[List[(List[Long], Double)], (List[Long], Double)]((List[Long](), 10000.0), 3)(
  (id, vd, msg) => (vd++msg).distinct.filter(l => (l._1 :: id).length == (l._1 :: id).distinct.length), //receive messages
  triplet => { Iterator((triplet.dstId, calcMsg(triplet))) }, //compute messages
  (a, b) => a++b //combine messages
)
```
Results

- Building graph nodes and edges are done using Map-Reduce in Spark
- The graph is built and traversed using GraphX
- Over 100M nodes and over 200M edges
- 80,000 unique values were reduced to 247
Summary

• **General framework:**
  • Blocking algorithm
  • String similarity
  • Build graph
  • Message passing for path finding
Summary

• **General framework:**
  • Blocking algorithm
  • String similarity
  • Build graph
  • Message passing for path finding

• This framework is being used for other applications including malicious actor tracking